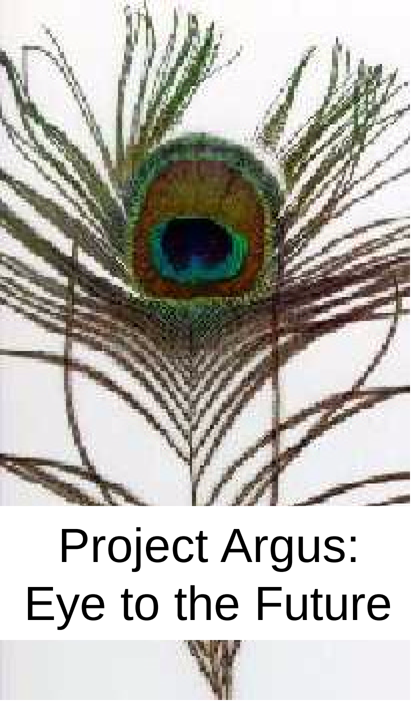


Project Argus:
Eye to the Future

Argus Prime

Adventures with ACT-R 5.0

Michael J. Schoelles

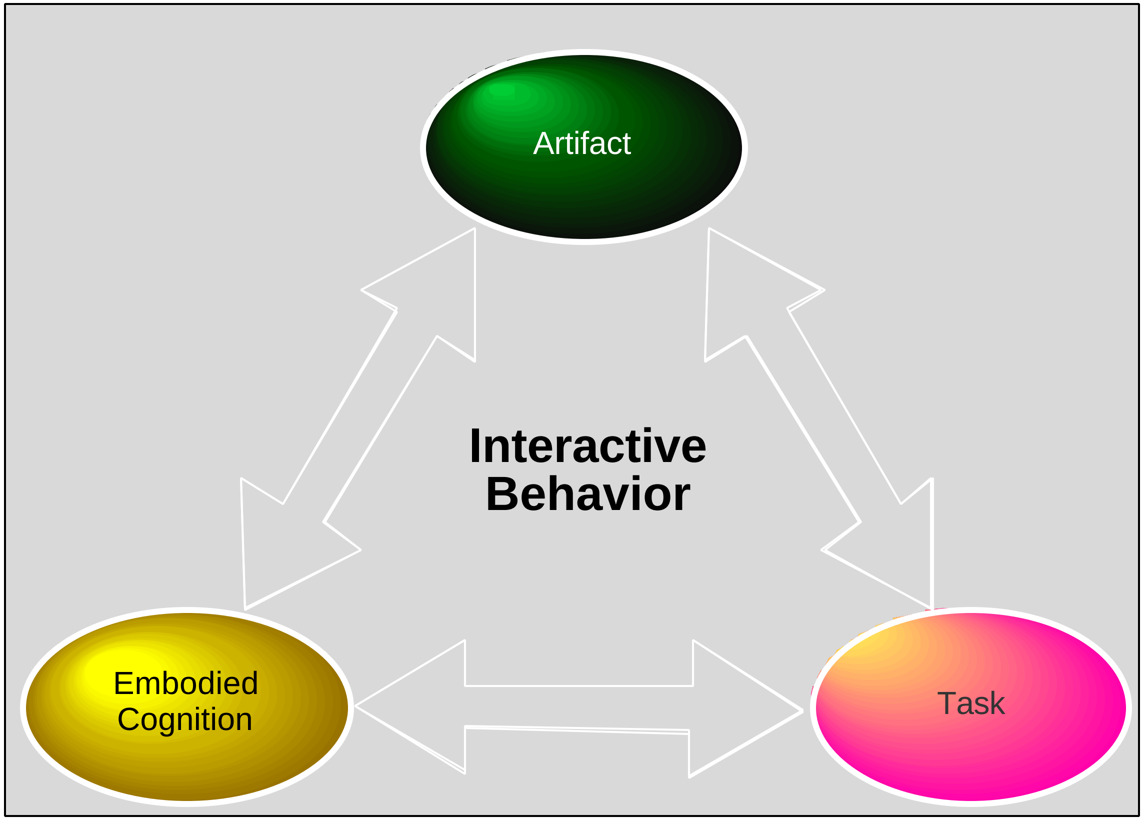
Wayne D. Gray

January 2002

Goals of This Talk

- **Review the Argus Prime Task**
- **Show matches and mismatches between performance of *Human Subjects* and *Model Subjects***
- **Describe the *Model Subjects* written in ACT-R 5.0**
  - **Proof of concept that ACT-R 5.0 should be the architecture of choice for complex HCI environments**
- **Discuss accommodation of *Model Subjects* to**
  - **Different interface conditions**
  - **User strategies**
- **Conclude: Engineering Approach to Building and Fitting Models of Interactive Behavior may use ACT-R 5.0 as its basis**

Perspective on Interactive Behavior



Argus Task

- **DEMO**

Data Collection

- Eye data (60 samples/second)
- Mouse data (60 samples/second)
- Mouse clicks
- Target data

Playback

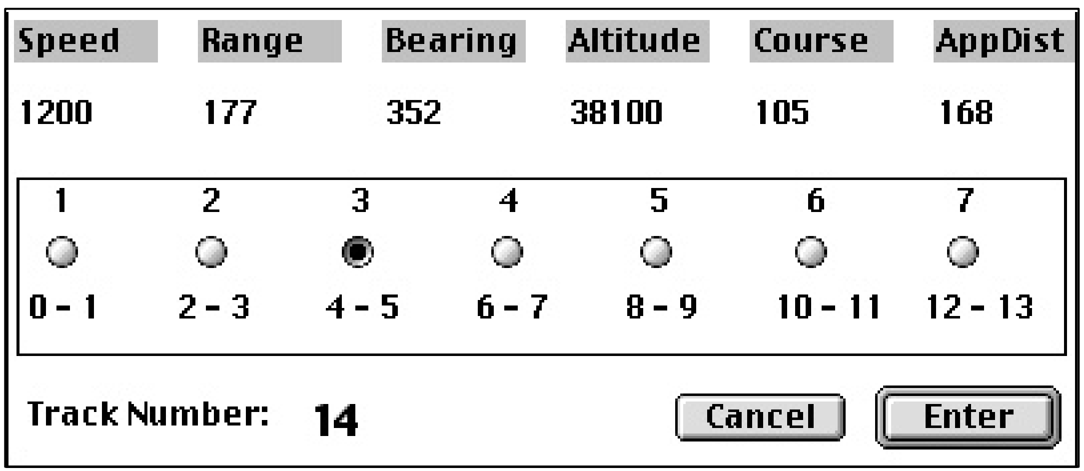
- **Demo**

We Will Discuss Data and Model from Two Studies

- **AP#4**
  - **No secondary tas**
  - **24 human subjects**
- **AP#5**
  - **Secondary tracking task**
  - **24 human subjects**
- **All other interface conditions were the same across studies**

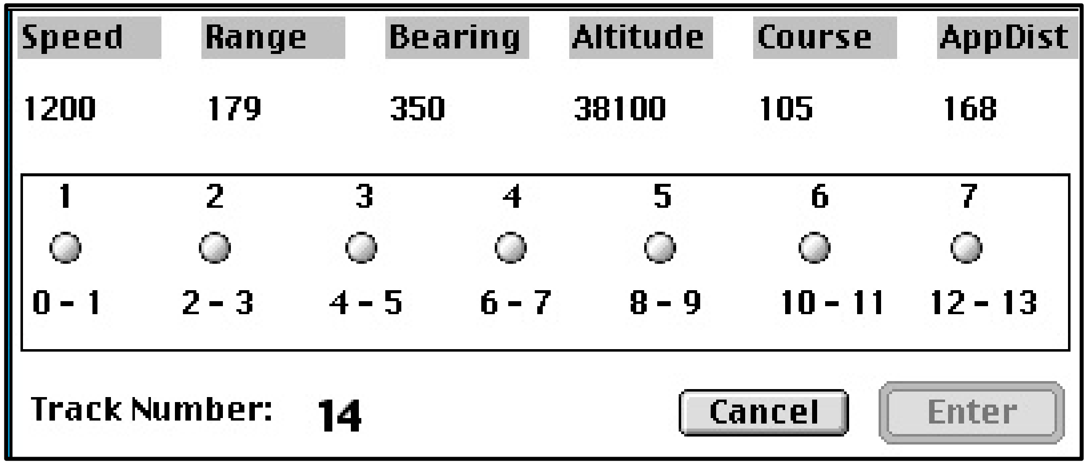
Within-Subject Interface Conditions

- Dot versus noDot
  - If target was already classified, then when it was reselected its threat value would be shown (with a dot) in the information window



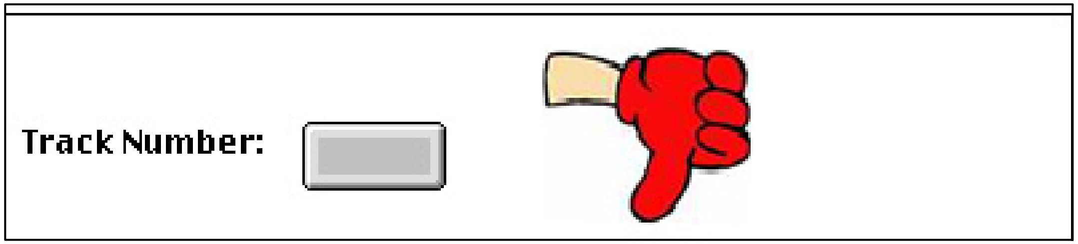
Within-Subject Interface Conditions

- **Dot versus noDot**
  - **Or not (noDot)**



Within-Subject Interface Conditions

- Feedback versus no-Feedback
  - Immediately following classification feedback either was available in a feedback window or was not



MODEL

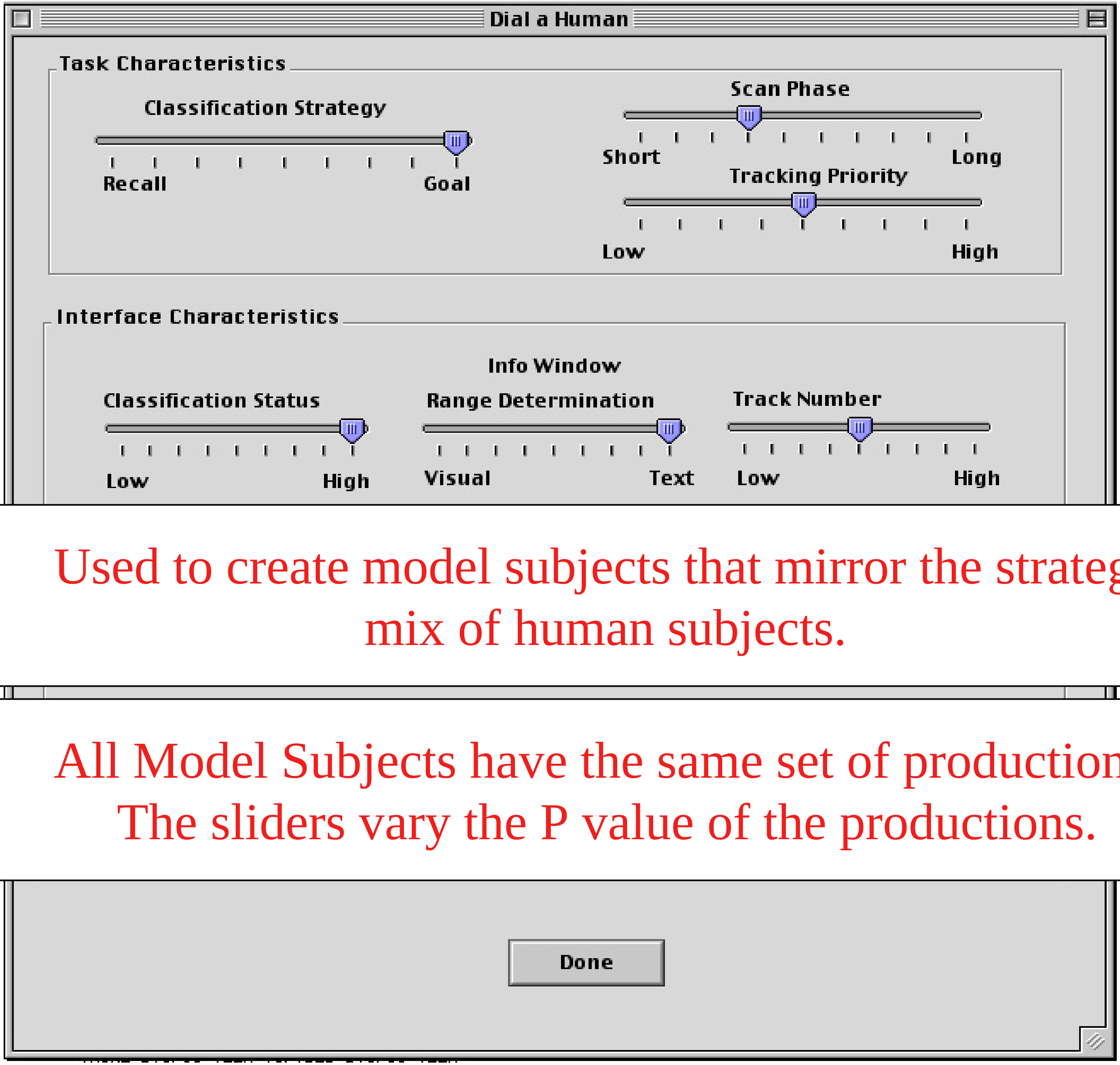
- **DEMO of one Model Subject doing AP#5**

Why Build Model Subject(s)

- **Between subject variability**
  - **Use of interface feature**
    - **Help**
    - **Feedback**
    - **Track Number**
    - **Filled in radio button**
  - **Visual Search Strategies**
  - **Cognitive operations**

How to Build a Model Subject

- Dial-A-Human
- Parameters of the Model Subject
- AP4 --> AP5



Used to create model subjects that mirror the strategy mix of human subjects.

All Model Subjects have the same set of productions. The sliders vary the P value of the productions.

User Parameters

- **User Interface Design**
  - **Create a mix that reflects user population**
- **User Interface Testing**
  - **Set for specific conditions**
- **Dial-A-Human implemented by changing the P value of relevant productions**

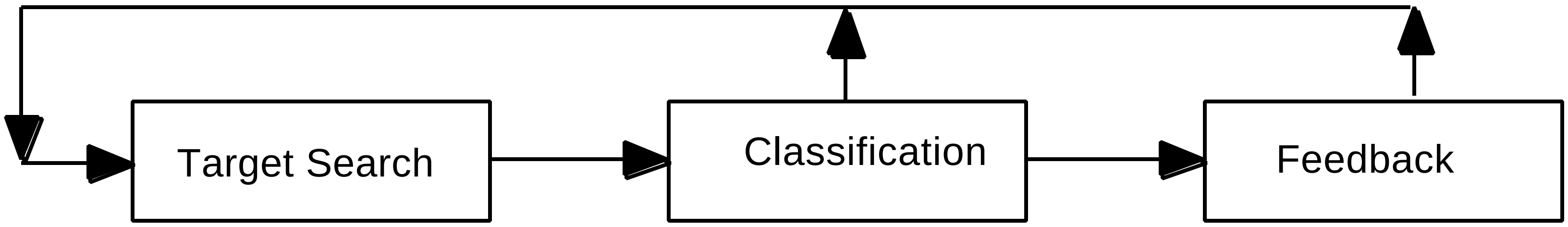
AP4 versus AP5 Model Subjects

- Single-task AP4 Model Subjects built to be matched to general mix of strategies found in AP4 Human Subjects
- Same Model Subjects used in AP5 for dual task
  - +1 parameter for tracking
  - No change to the classification task productions and parameters used by these Model Subjects for AP4

Modeling Single versus Dual Task Performance

- **AP4 -- Single Task Performance**
- **AP5 -- Dual Task Performance**
  - **Interleaving of Tracking with Classification Task**

Argus Unit Tasks - single task

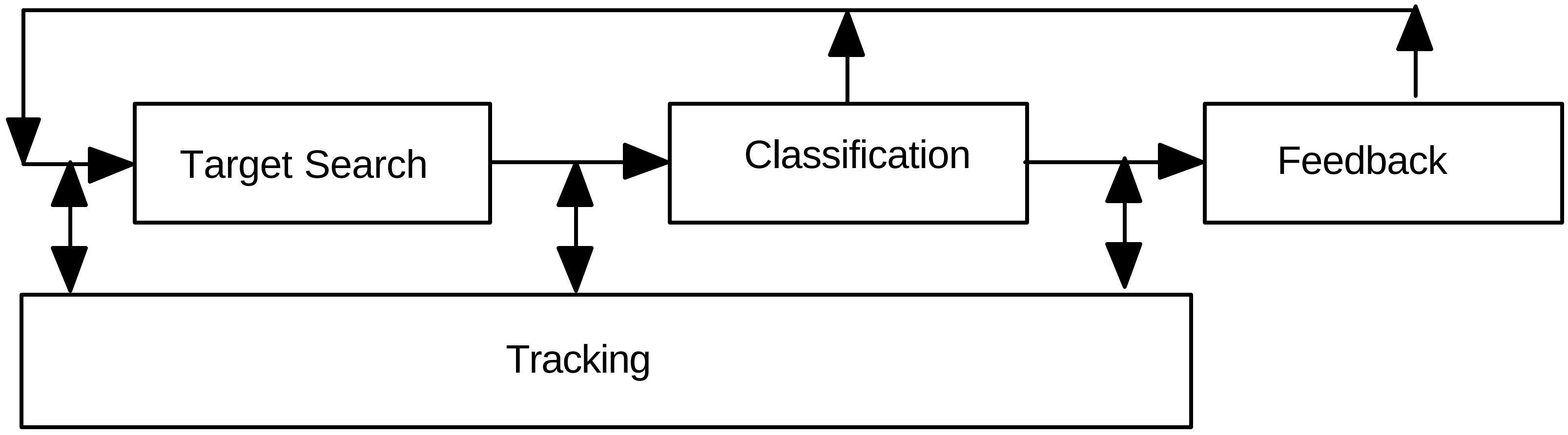


Dual Task Environment

- **Tracking task**
  - **A perceptual/motor task**
  - **Increase workload**
- **Can model predict degradation of performance on primary task in dual task condition?**
- **Used same Model Subjects**

Tracking Unit Task

- **The model switches to the tracking task at unit task boundaries**



Tracking Unit Task

- During the Classification task, the Model checks, between unit tasks, to determine whether it should switch to the tracking task
- Tests to determine
  - If the tracking cursor has changed color (blue to yellow or to red)
  - Difference in score between classification task and tracking task
- Stays switched until tracking cursor changes back to blue

Characteristics of Computational Models of Embodied Cognition in a Dynamic Task

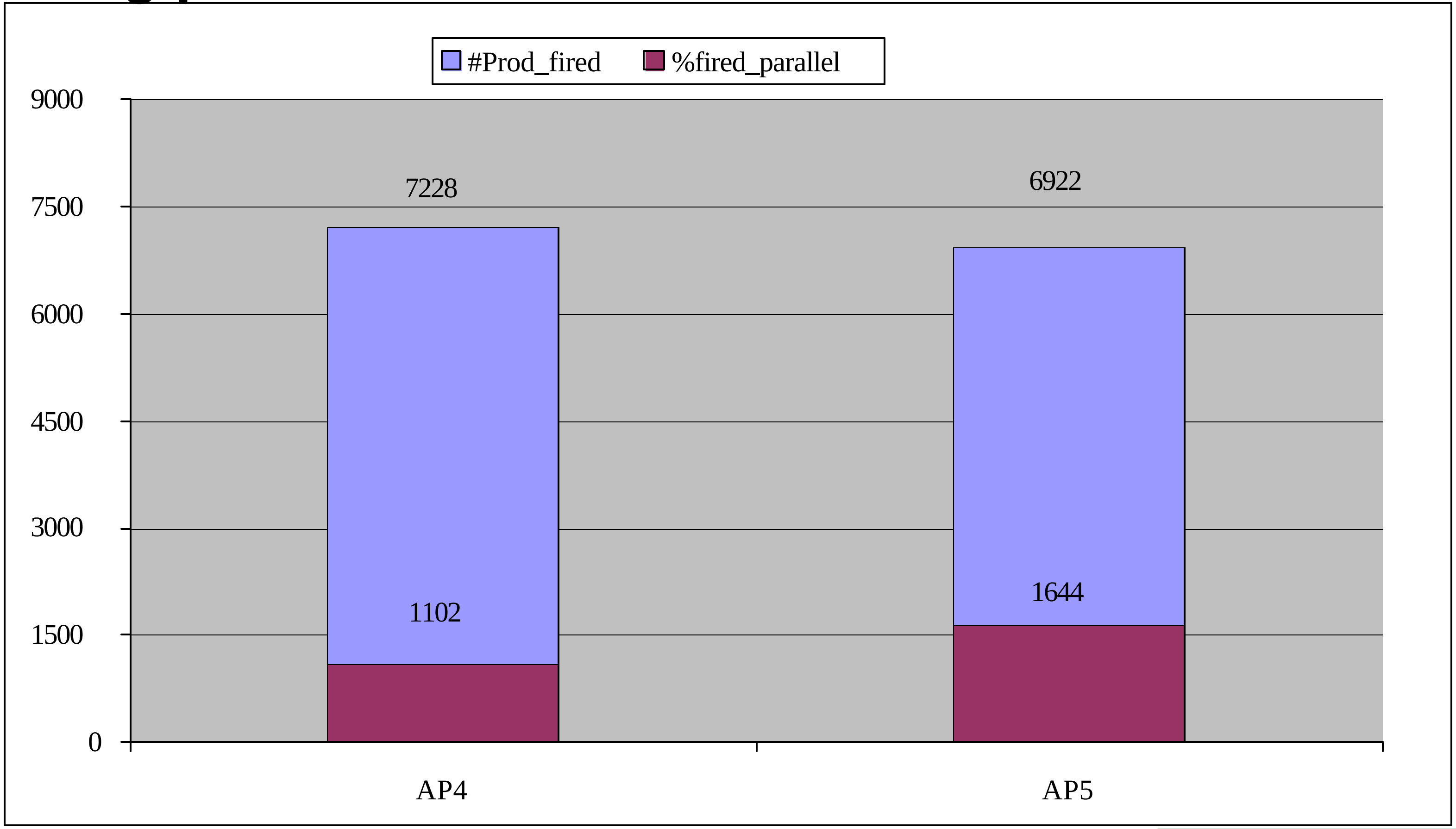
- **Production counts**
- **Production Firings**
- **Buffer actions**
- **Declarative Memory Elements created as the Model Subject performs a scenario**
- **Hot off the press results, to provide a flavor of the complexity of the model, not (yet) as a detailed comparison between model runs of AP4 versus AP5**

Model Structure

- 243 Productions
  - 96 change the goal buffer
  - 61 retrievals
  - 43 feature search
  - 22 attention shifts
  - 33 motor operations
- 33 productions initiate parallel operations
  - 2 buffer changes on RHS

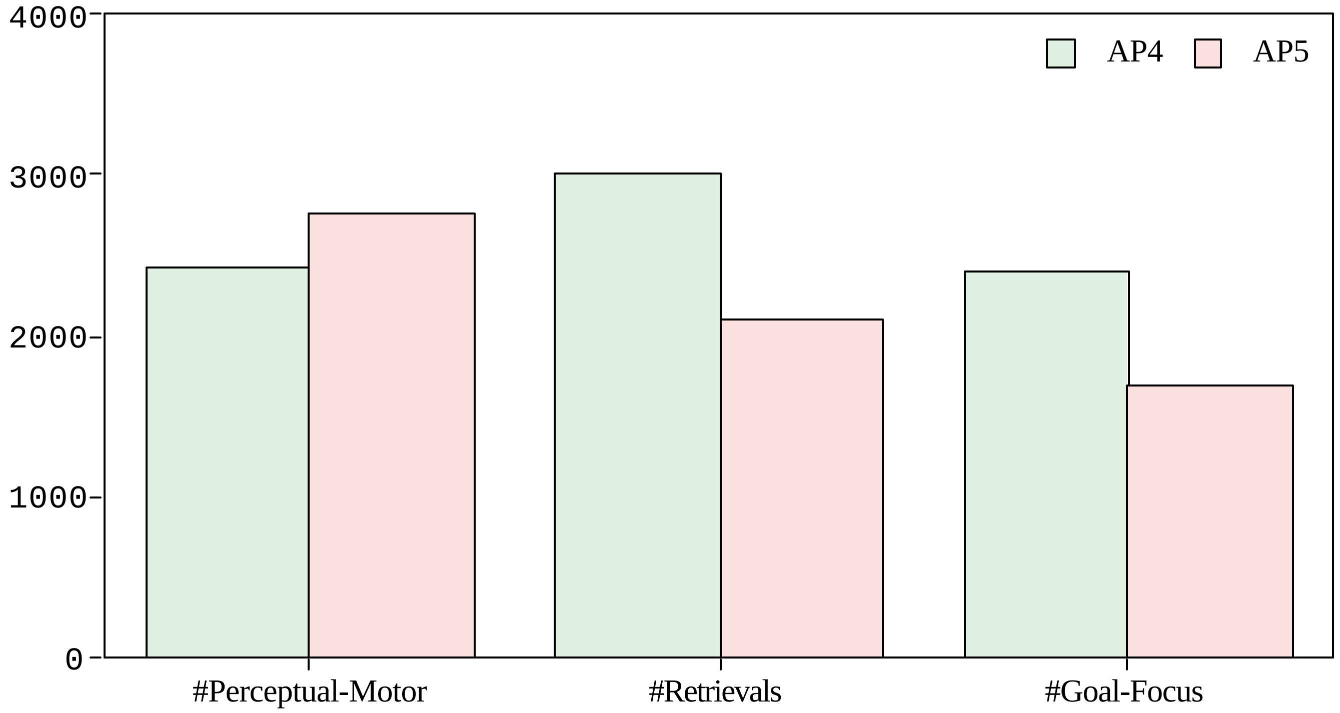
Model Execution

- **Mean number productions fired and mean number firing parallel actions**



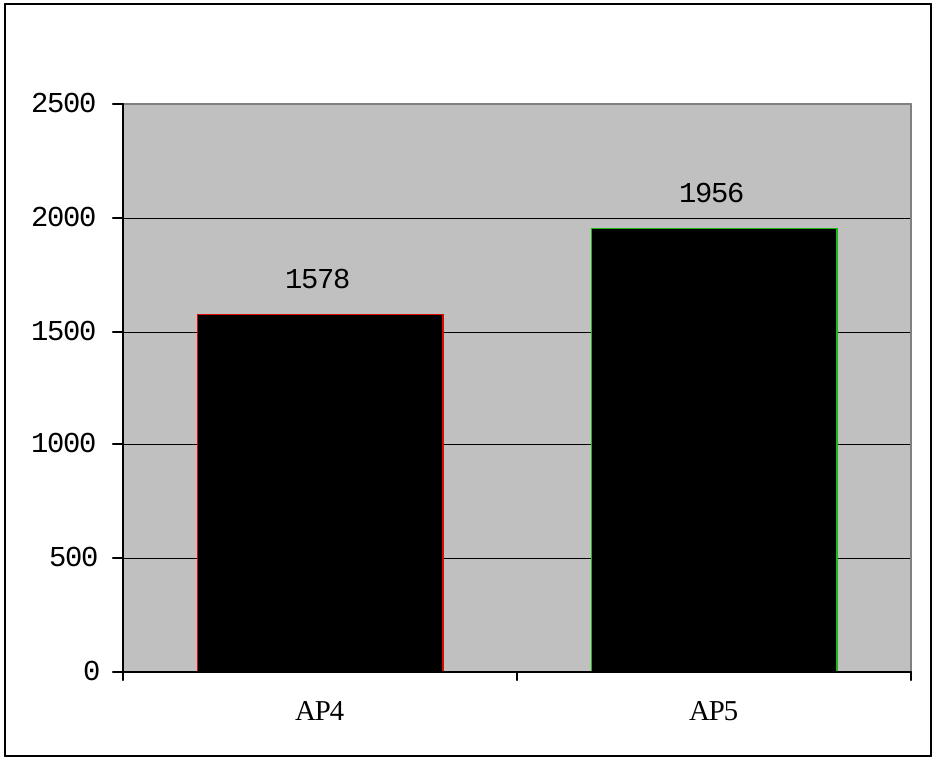
Model Execution

- **Mean number of buffer actions on RHS**



Model Execution

- **Mean number of new Declarative Memory Elements created**



Modeling Dynamic Strategies

- Target Selection Unit Task
  - Attentional Marking
    - (I saw that already? Or did I?)
  - Target Search
- Classification Unit Task
- Feedback Unit Task

Target Selection

- **Multiple visual search strategies**
  - **Scan, border, nearest, cluster, random, priority**
- **Conflict set resolution**
  - **Lisp function that considers multiple features of environment**
  - **Last location**
- **Each strategy implemented by feature search**
- **Raises important issues for ACT-R regarding attentional marking**

Attentional Marking (FINST)

- **Provides limits on the Vision Module's memory for what has and has not been attended**
  - **Unlimited memory size and duration unrealistic**
- **Why is this important in Argus Prime?**
  - **Targets are often revisited**
  - **Degree of revisitation varies by**
    - **Interface condition**
    - **Search strategy**
  - **Requires constant shifts in visual attention, many information fields, 20 targets, etc**
- **Hard to control both size and duration**

Target Search Production

```
(defp search-targ-cluster-in-seg
  =goal>
        isa            cluster-in-seg-search
     step nil
  =visual-state>
        isa            module-state
     modality          free
  =retrieval>
        isa            search-parameters
        seg            =seg
==>
  +visual-location>
        isa            visual-location
        attended       nil
     value             =seg
        nearest        current
        kind           screen-targ
  =goal>
        step           check-result)
```

Classification Unit Task

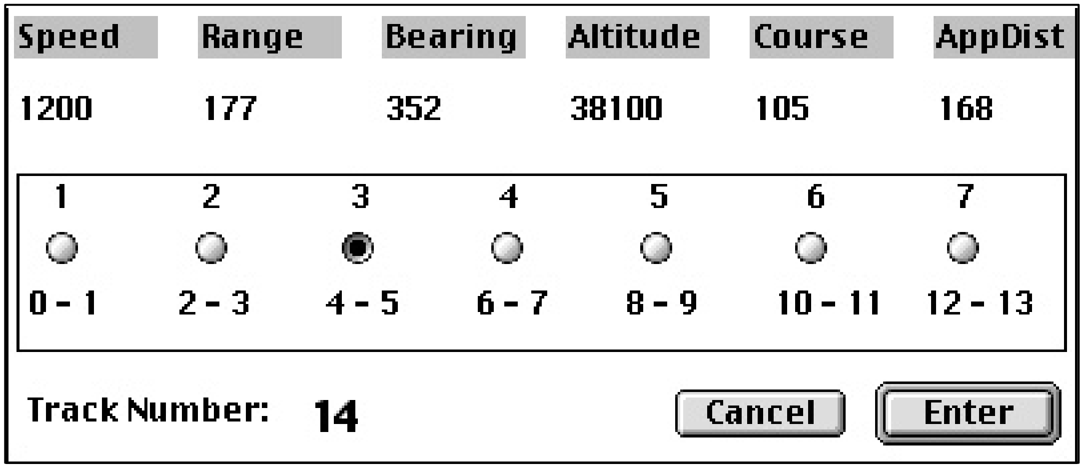
- **Classification calculation strategies**
  - **Intermediate results retrieved from declarative memory**
  - **Intermediate results stored in goal**
  - **Help facility**

Model Subjects vs Human Subjects

- **Overall performance**
- **For each Unit Task look at variety of measures of Quantity and Duration of Interactive performance**
- **19 Quantity Measures & 11 Duration Measures have been looked at to date**
  - **We will only present the major ones**
- **Also -- we will focus ONLY on differences between Model Subjects and Human Subjects; not on differences between within-subject conditions**

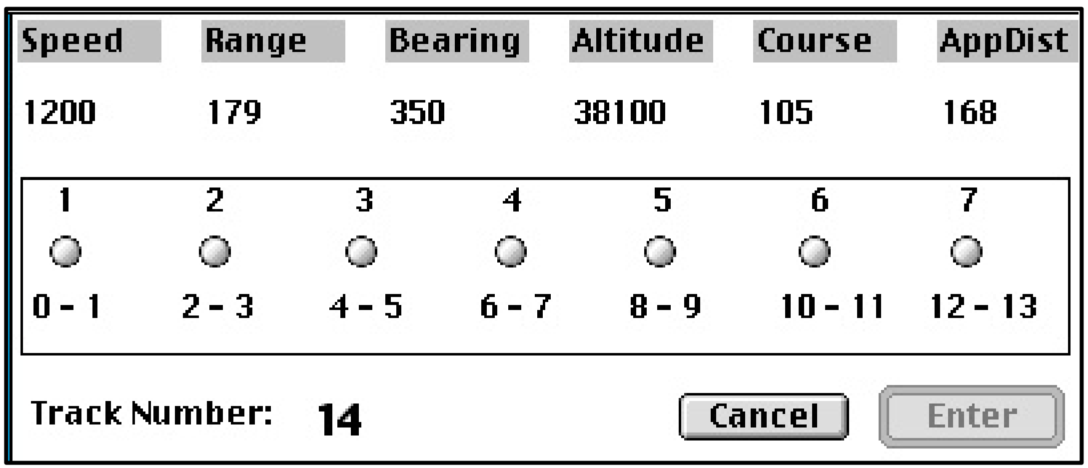
Within-Subject Conditions

- **Dot versus noDot**
  - **If target was already classified, then when it was reselected its threat value would be shown (with a dot) in the information window**



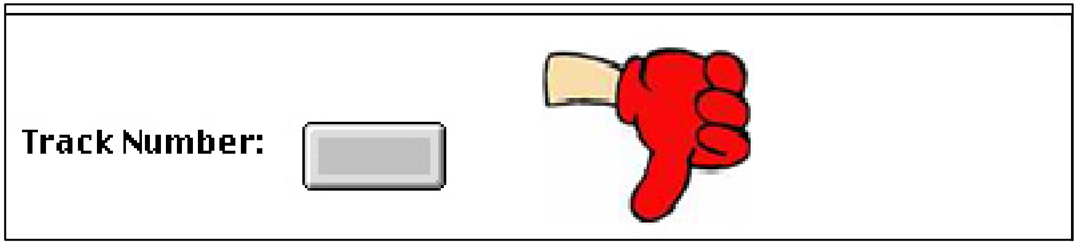
Within-Subject Conditions

- **Dot versus noDot**
  - **Or not (noDot)**

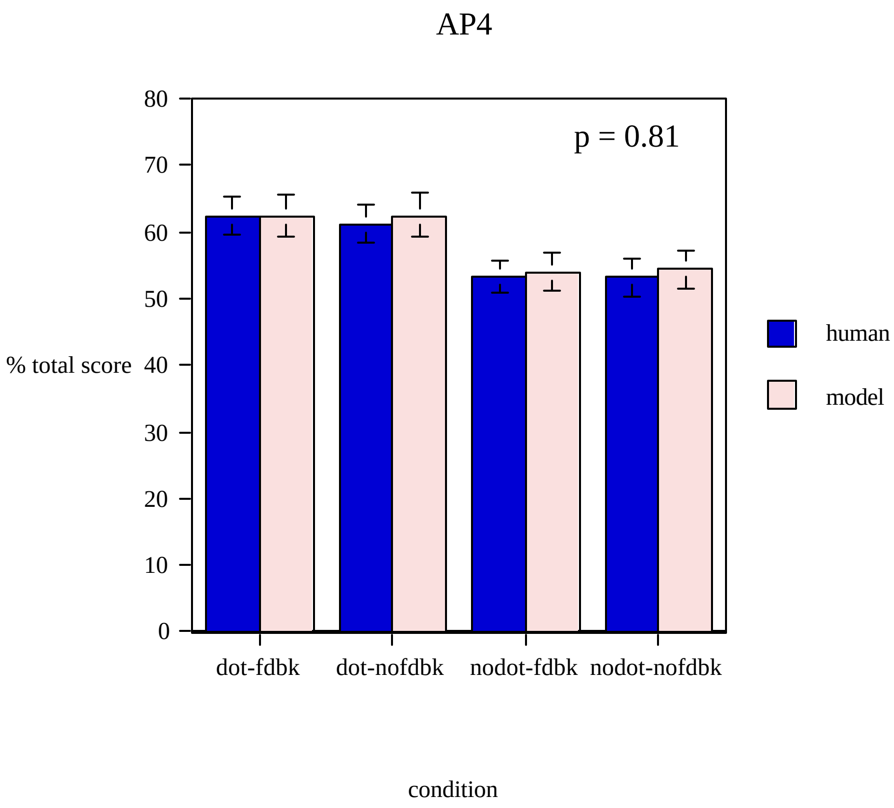


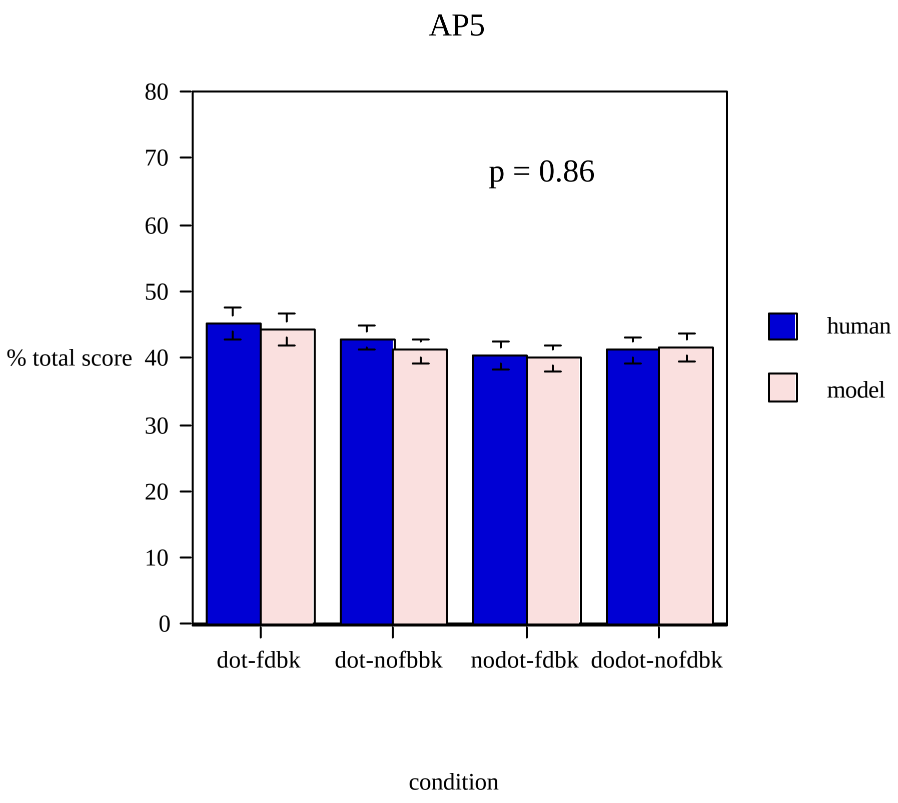
Within-Subject Conditions

- **Feedback versus no-Feedback**
  - **Immediately following classification feedback either was available in a feedback window or was not**

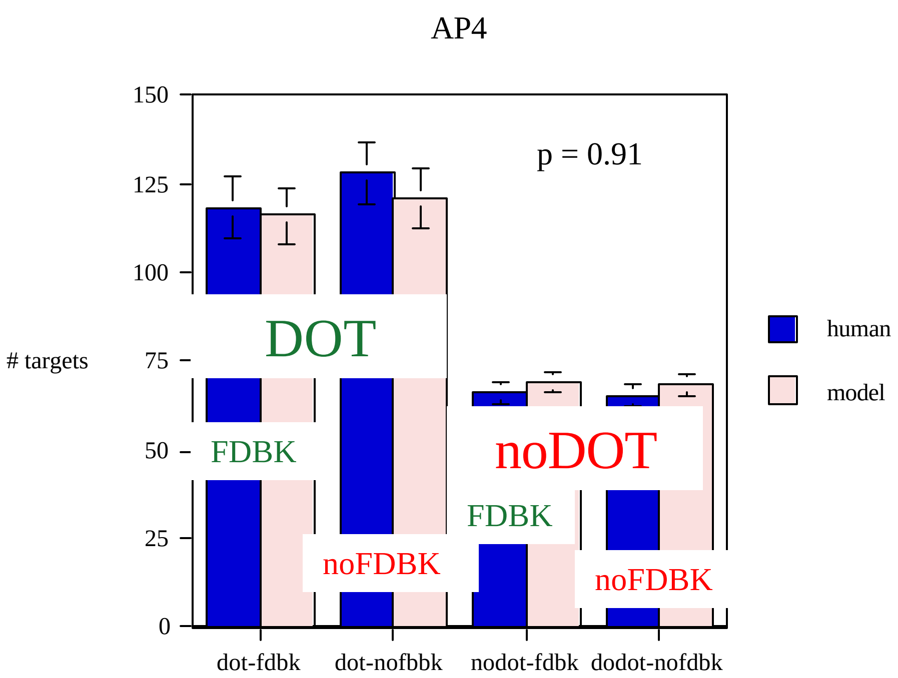


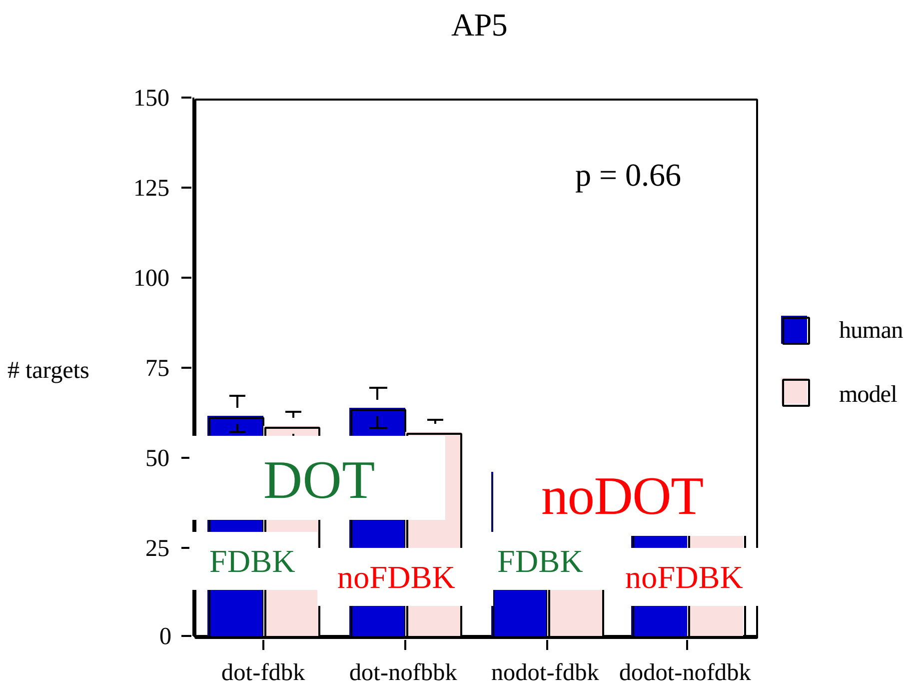
Total Score- Classification Task



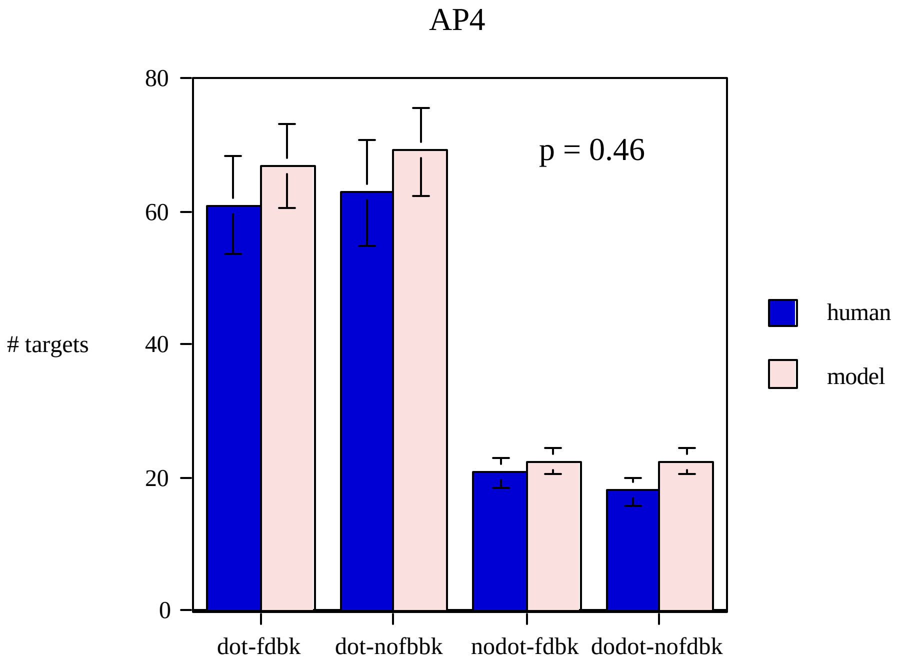


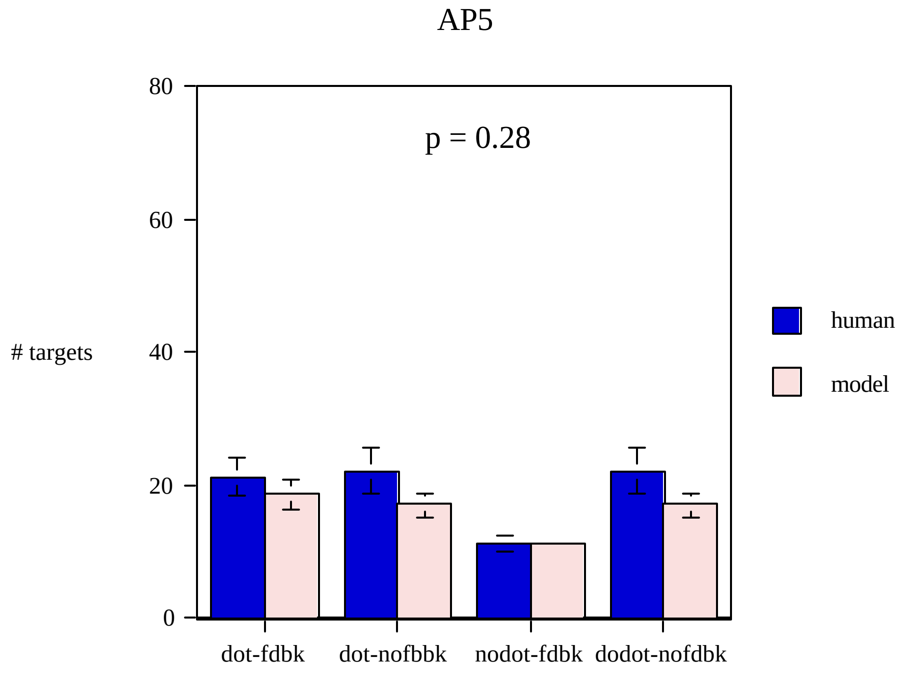
Total # of Target Selections



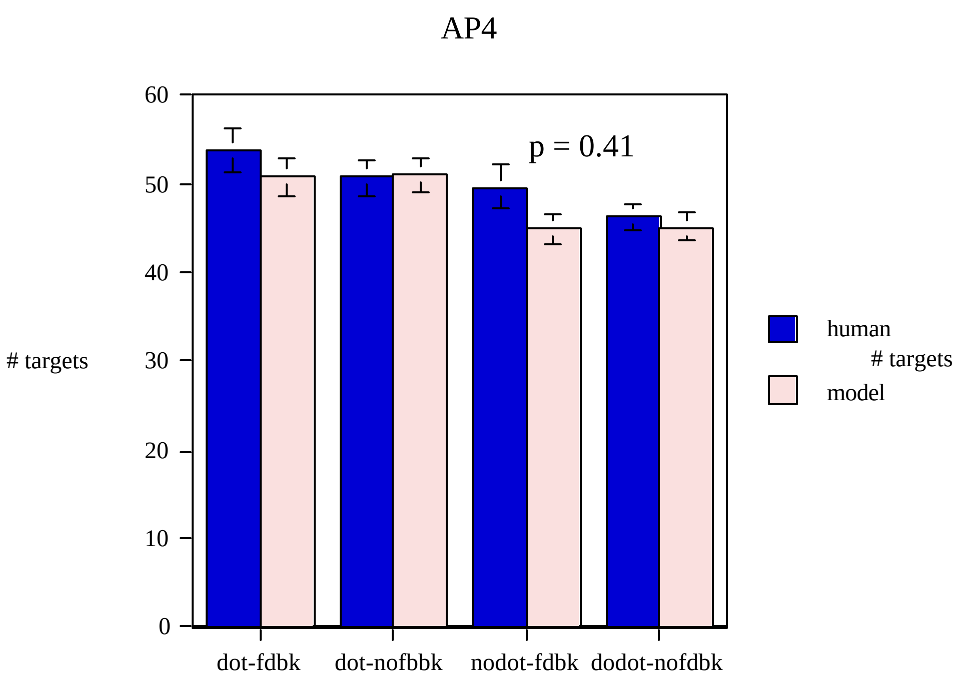


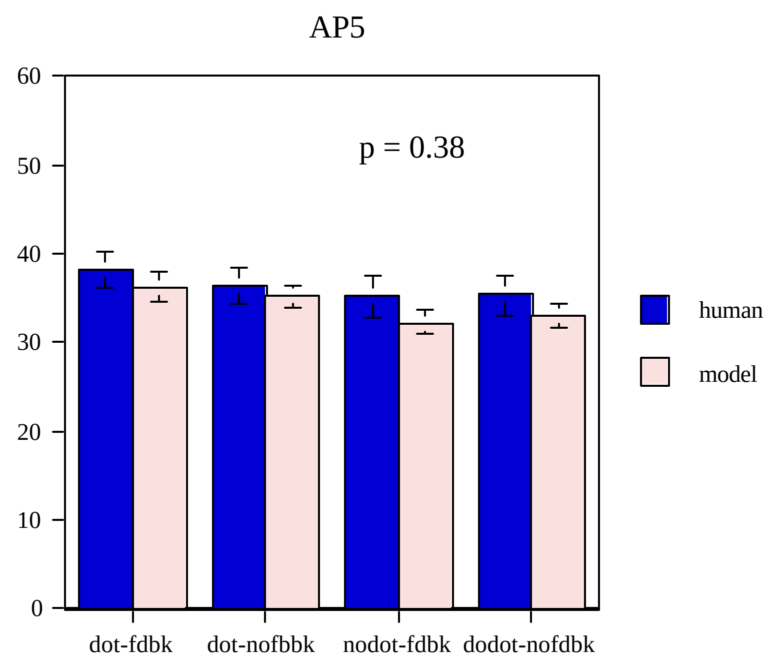
Re-selecting Already Classified Targets



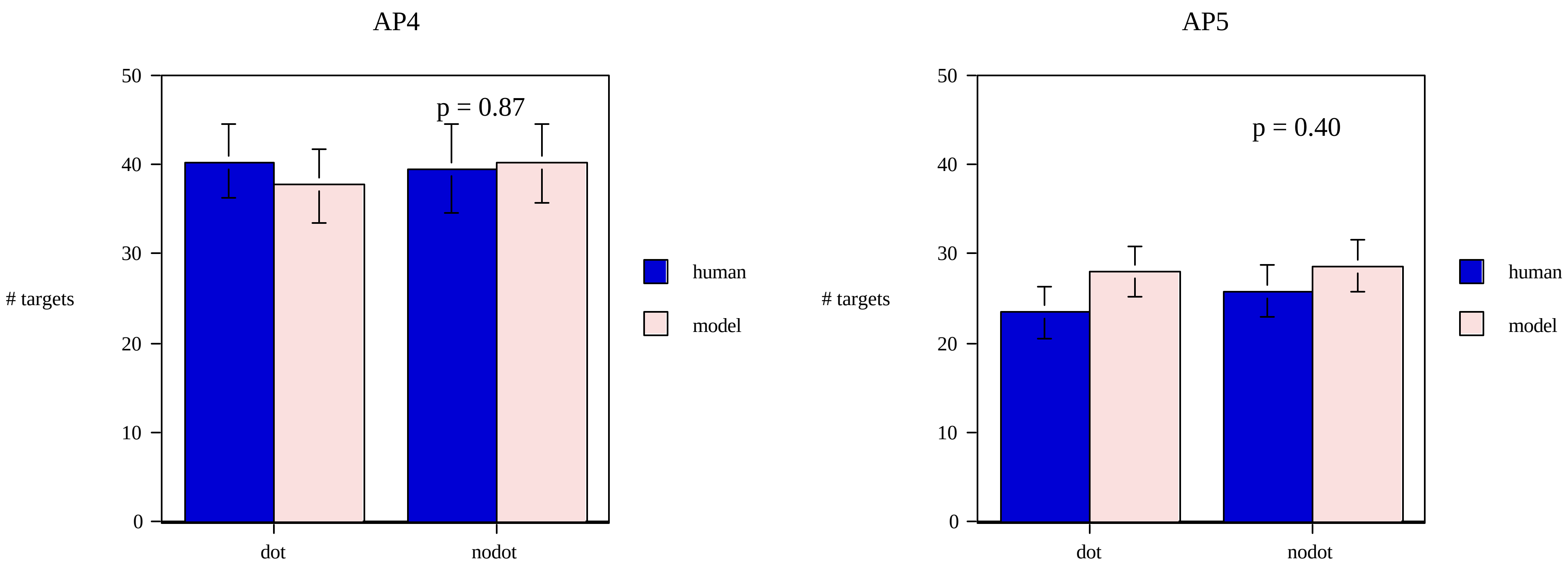


Total # of Classifications



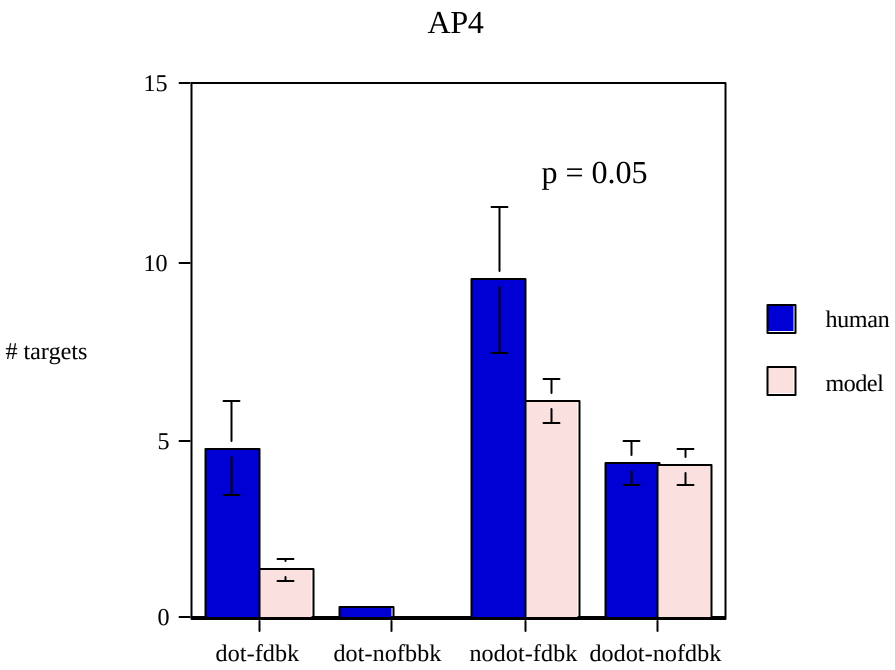


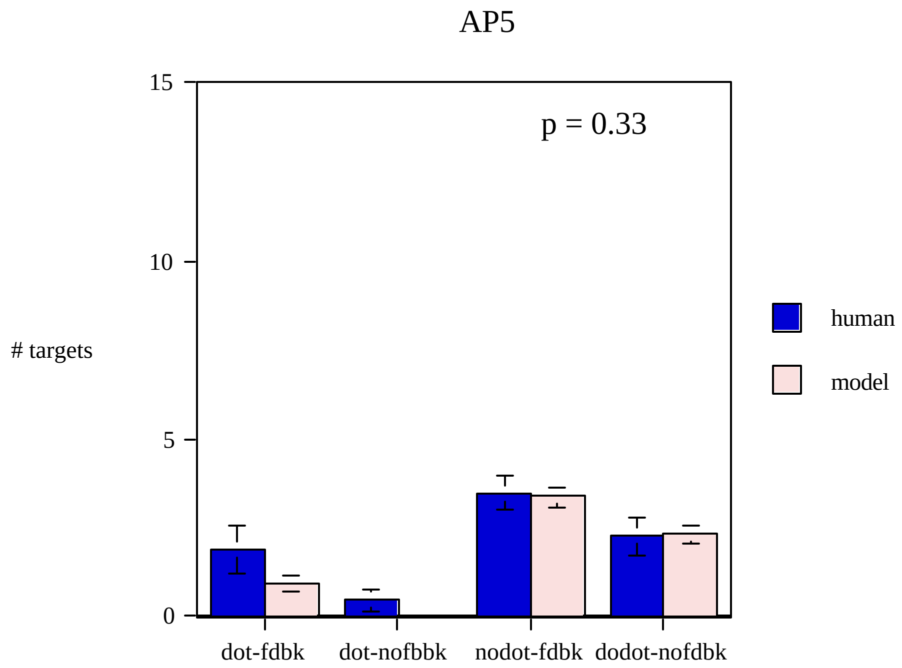
of Feedback Processing



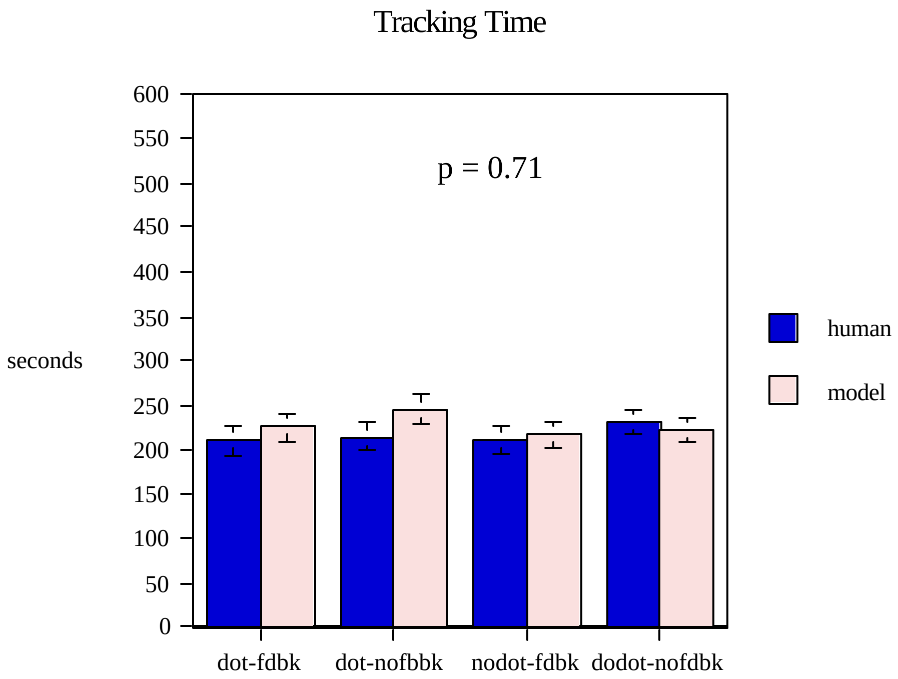
George Mason University
Human Factors & Applied Cognitive Program

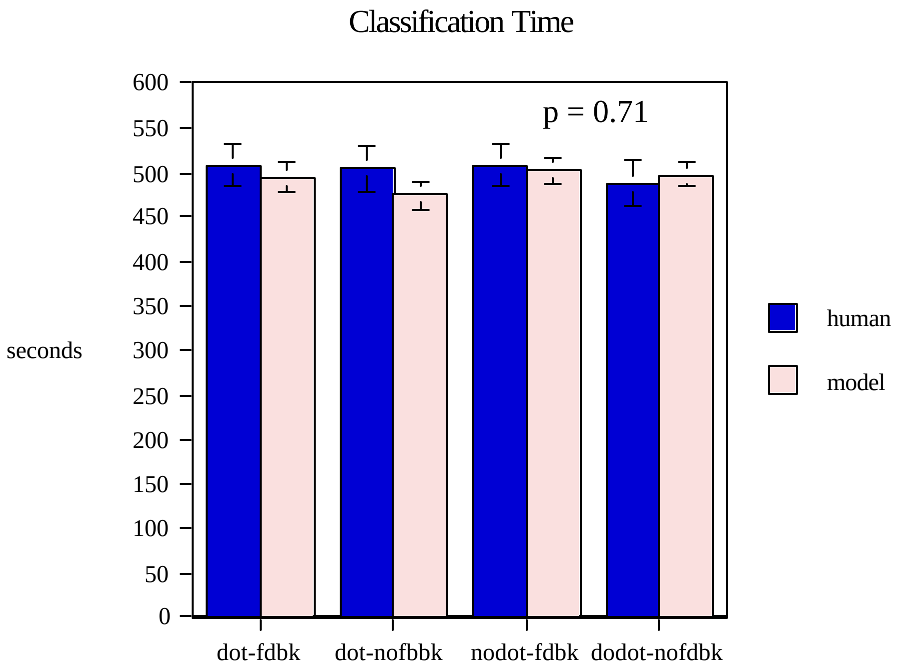
Total Number Reclassified





AP5 Tracking Time versus Classification Time





George Mason University
Human Factors & Applied Cognitive Program

Unit Task Performance Summary

Unit task	# Performanc	# no sig. diff. AP4	# no sig. diff. AP5
Target Selection	5	5	4
Classification	4	4	3
Feedback	3	2	3

Performance measures for Target Selection:

Total targets selected

Number of unclassified targets selected

Number of already classified targets selected

Number of times target was selected but then ignored

Number of targets selected (without use of feedback)

Unit Task Performance Summary

Unit task	# Performanc	# no sig. diff. AP4	# no sig. diff. AP5
Target Selection	5	5	4
Classification	4	4	3
Feedback	3	2	3

Performance measures for Classification:

- Total classifications
- Total correct
- Number initially incorrect
- Number classifications (no help)

Unit Task Performance Summary

Unit task	# Performanc	# no sig. diff. AP4	# no sig. diff. AP5
Target Selection	5	5	4
Classification	4	4	3
Feedback	3	2	3
Tracking	7	n/a	6

Unit task	# Timing	# no sig. diff. AP4	# no sig. diff. A
Target Selection	3	1	0
Classification	2	1	2
Feedback	2	0	0
Tracking	4	n/a	4

Conclusion

- Arguably the worlds most complex ACT-R 5.0 model
- Model Subjects based on AP4 predict performance on Classification Task used in AP5
- Engineering approach

ACT-R 5.0 Discussion Issues

- Downgrading of goal stack
  - Subgoals
  - Parameter passing
- LISP Package for ACT R 5.0 to avoid symbol conflicts
- Dynamic updating of iconic memory
- Eye movements to a location with no object present
  - (EMMA)
- Variable Types
  - Problem of reuse vs re-creation of productions for low level, fundamental actions such as converting screen-symbols to numeric values.
  - Sharing of idioms (procdures)

Subgoals & Parameter Passing

- ACT-R factorial example
- ACT-R 5.0 implementation
- Argus model idiom

```
(chunk-type plus-fact addend adder sum)
(chunk-type times-fact multiplicand multiplier product)
(chunk-type factorial argument result)
(add-dm
 (0+1=1 isa plus-fact addend 0 adder 1 sum 1)
 (1+1=2 isa plus-fact addend 1 adder 1 sum 2)
 (2+1=3 isa plus-fact addend 2 adder 1 sum 3)
 (1*1=1 isa times-fact multiplicand 1 multiplier 1 product 1)
 (1*2=2 isa times-fact multiplicand 1 multiplier 2 product 2)
 (2*3=6 isa times-fact multiplicand 2 multiplier 3 product 6)
 (fact3 isa factorial argument 3))

(goal-focus fact3)
```

```
(p base
  =goal>
     isa factorial
     argument 0
==>
  !output! (The factorial of 0 is 1)
  =goal>
     result 1
  !pop!)

(p recurse
  =goal>
     isa factorial
     argument =n
     result nil
  =fact>
     isa plus-fact
     addend =n-1
     adder 1
     sum =n
==>
  !output! (Subgoaling the factorial of =n-1)
  =subgoal>
     isa factorial
     argument =n-1
     result =result
  =goal>
     result =result
  !push! =subgoal)
```

```
(p compute
  =goal>
     isa factorial
     argument =n
     result =fact-n-1
  =fact>
     isa times-fact
     multiplicand =fact-n-1
     multiplier =n
     product =fact-n
==>
  !output! (Computing the factorial of =n as =fact-n)
  =goal>
     result =fact-n
  !pop!)
```

```
(chunk-type factorial argument result step next)
(P base
  "base"
  =goal>
    ISA         factorial
    argument    0
    next        =parentgoal
==>
  !output!      ( the factorial of 0 is 1)
  =parentgoal>
    ISA         factorial
    result      1
  +goal>        =parentgoal)

(P recurse
  "recurse"
  =goal>
    ISA         factorial
    argument    =n
    result      nil
    step        nil
==>
  =goal>
    step        retrieving
  +retrieval>
    ISA         plus-fact
    adder       1
    sum         =n)

(P recurse-retrieve
  "recurse-retrieve"
  =goal>
    ISA         factorial
    argument    =n
    result      nil
  =retrieval>
    ISA         plus-fact
    addend      =n-1
    adder       1
    sum         =n
==>
  !output!      ( subgoaling the factorial of =n-1)
  =goal>
    step        nil
  +goal>
    ISA         factorial
    argument    =n-1
    next        =goal)

(P compute
=goal>
    ISA         factorial
    argument    =n
    result      =fact-n-1
    step        nil
==>
  =goal>
    step        computing
  +retrieval>
    ISA         times-fact
    multiplicand =fact-n-1
    multiplier  =n)

(P compute-retrieve
=goal>
    ISA         factorial
    argument    =n
    result      =fact-n-1
    next        =parentgoal
    step        computing
  =retrieval>
    ISA         times-fact
    multiplicand =fact-n-1
    multiplier  =n
    product     =fact-n
==>
  !output!      ( computing the factorial of =n as =fact-n)
  =parentgoal>
    ISA         factorial
    result      =fact-n
  +goal>        =parentgoal)
(P compute-done
=goal>
    ISA         factorial
    argument    =n
    result      =fact-n-1
    next        nil
    step        computing
  =retrieval>
    ISA         times-fact
    multiplicand =fact-n-1
    multiplier  =n
    product     =fact-n
==>
  !output!      ( computing the factorial of =n as =fact-n)
  =goal>
    step        done)
```

Idiom used in Argus Model

```
(deftype parentgoal result)
(deftype subgoal next)
(deftype (parent&subgoal (:include parentgoal)) next)
(deftype (chk-tot-% (:include subgoal)))
(deftype (select-target (:include parentgoal)) step init-targ reclass loc)
(deftype (compare-% (:include parent&subgoal)) step class track)
(defp chk-tot-%-3
 =goal>
       isa        chk-tot-%
       step       look-at-tot%
   next   =newgoal
 =visual-state>
       isa        module-state
   modality       free
 =visual>
       isa        Fdbk-txt
       val        =txt
==>
 =newgoal>
        isa       parentgoal
       result     =txt
 +goal>   =newgoal)
```

Model Parameters

- Perceptual
  - Visual attention latency = 0.85
    - Source: default
  - Number of FINSTS = 100
  - FINST span = 20
  - Onset span = 5.0
- Motor
  - defaults

Model Parameters

- **Cognitive**
  - **Base level learning (:bll) = 0.5**
    - **Source: de facto default**
  - **Activation noise (:ans) = 0.25**
    - **Source: near de facto default**
  - **Latency factor (:lf) = 0.05**
    - **Source: ACT-R list**
  - **Expected gain noise (:egs) = 0.20**
    - **Source: ??**